Visual Mining and Statistics for a Turbofan Engine Fleet

Jérôme Lacaille, Etienne Côme

Visual Mining and Statistics for a Turbofan Engine Fleet

Jérôme Lacaille
Snecma
Etablissement de Villaroche sud
77550 Moissy-Cramayel Cedex, France
+33 1 60597024
jerome.lacaille@snecma.fr

Etienne Côme
SAMM
Université Paris 1 Panthéon Sorbonne
90, rue de Tolbiac
75013 Paris, France
etienne.come@univ-paris1.fr

Abstract—Snecma, as a turbofan manufacturer, needs to deal with a wide fleet of more than thousands of engines. Every day, data from aircraft engines are broadcasted to the ground. Some airlines companies rely on their engine manufacturer to control the engines' behavior and help prepare for maintenance scheduling. The goal of the manufacturer is to detect abnormalities to help schedule maintenance operations. The advantage of the manufacturer as MRO operator is the registered memory of all past events that appears on its fleet of engines. If one opens the possibility to look in this huge amount of data for corresponding similar behaviors, which may have append in the past (for all engines of all customer companies), it becomes possible to make some targeted statistics of the future.

This paper describes an algorithm to help engineers looking at some wear indicators and proposes a new full automatic method able to fetch information stored from many years of operations. This methodology is based on mathematic foundations. It uses a formalization of engine trajectories; creates a metric space where it becomes possible to compare time intervals of the evolution of engines. A generic maintenance application uses this methodology and finds in the fleet database engines that were similar in behavior to build future statistics. [1] [2]

TABLE OF CONTENTS

GLOSSARY

ACARS	Aircraft Communication Addressing and Reporting System
BMU	Best Matching Unit
EGT	Exhaust Gas Temperature
FF	Fuel Flow
GLM	Generalized Linear Model
MRO	Maintenance Repair and Overhaul
MSE	Mean Square Error
N1, N2	Fan and core speeds
PCA	Principal Component Analysis
RLS	Recursive least Square
SOM	Self_Organizing Map

1. INTRODUCTION

During normal operation, an aircraft regularly sends small ACARS (Aircraft Communication Addressing and Reporting System) messages to the ground. More messages are sent if the FADEC (Full Authority Digital Engine Control) engine controller detects unsuspected events. The content of all theses messages is stored in a database. It consists mainly of measurements taken on each engine.

Today's state of the art is the simultaneous analysis of three of these measurements: the core shaft speed (N2), the fuel flow (FF) and the exhaust gas temperature (EGT). Looking at temporal curves built from successive flights, the engineers are able to recognize some classical patterns. Such tool involves manual exploration of the data and transmission of knowledge between engineers. Moreover, expert engineers are specialized for specific fleets of aircrafts that always did the same kind of missions. For example, the patterns corresponding to aircraft flying over sea are different from those coming from planes that are staying over the land.

Our new solution is based on a non-supervised classification of engine states. This classification use the self-organizing maps (SOM) algorithm to project the current engine state on a 2D map that may be colored according to known defective behaviors. The observation of an engine state trajectory that converges to such known defective paths on a 2D map is a very useful and straightforward application. But the implementation of a distance between trajectory intervals on the map let us detect other engines that may have followed a similar path in the past. This is an obvious way to build future statistics.

2. INPUT DESCRIPTION

The transmission protocol of data from aircraft to the ground uses a satellite link but the message should be limited to 4Kb. However it is possible to sent a snapshot of measurements taken from the aircraft computer and both engine controllers. Two messages are always present during a flight, one at take-off and the other during cruise.

[1] 978-1-4244-7351-9/11/$26.00 ©2011 IEEE.
[2] IEEEAC paper #1042, Version 2, Updated December 22, 2010

Maintenance engineers use to look at the temporal behavior of three measurements for each engine (N2, EGT and FF). We used two more of them for the prototype of our new system: the static pressure (PS3) and temperature (T3) after compression and before combustion chamber. Those last measurements gives information about the compressor behavior and help identify difference between compressor and turbine degradation.

More than just adding two variables, we also analyze all take-off and cruise measurements together and we use aircraft and other engine information to normalize the data. The table below shows measurements selected for analyze and those kept for context normalization.

Table 1: Input measurements.

Name	Description
Index information	
AC_ID	*Aircraft ID*
ESN	*Engine Serial Number*
FL_DATE	*Flight Date*
Context information	
TAT	*External temperature*
ALT	*Altitude*
AIE	*Anti Ice Engine*
AIW	*Anti Ice Wings*
BLD	*Bleed valve position*
ISOV	*ECS Isolation Valve Position*
VBV	*Variable Bleed Valve Position*
VSV	*Variable Stator Vane Position*
HPTACC	*High Pressure Turbine Active Clearance Control*
LPTACC	*Low Pressure Turbine Active Clearance Control*
RACC	*Rotor Active Clearance Control*
ECS	*Environmental Control System*
TLA	*Thrust Lever Angle*
N1	*Fan Speed*
XM	*Mach Number*
Monitored measurements	
N2	*Core Speed*
FF	*Fuel Flow*
PS3	*Static pressure after compression*
T3	*Temperature after compresion*
EGT	*Exhaust Gas Temperature*

The measurements should be acquired regularly over the time. We automate the process with a data link between the satellite reception station and our database. However this information belongs to client companies, which should not inform their MRO about all small maintenance operations (mainly under the wing). As the components information is forwarded to another database we may discover some unexpected events before the synchronization is done. This source of confusion should be avoided.

3. METHODOLOGY

The algorithm uses a pretreatment of the data: the measurements are first normalized according to flight information. This will get rid of airplane attitude, dependency to pilot commands and any flight specific context. The residual noise is removed after taking care of some brutal changes observed on the data curves. These changes may represent real maintenance events. Once cleaned, the observations are classified with a self-organizing map. The algorithm builds a classification that takes continuity of observations into account.

Suppression of acquisition context

The acquisition context is defined by some measurement describing the flight, the airplane attitude and other environmental information. Even if the snapshot acquisition time is selected during a stable flight phase, some differences may appear. For example the first flight of the day will differ a little from the others at takeoff because the engine was cold. Weather conditions may be different; the pilot is not the same, etc. All these little things, cumulated, finally impact the engine measurements.

This following image, already presented in [14] shows the impact of normalization process on the EGT.

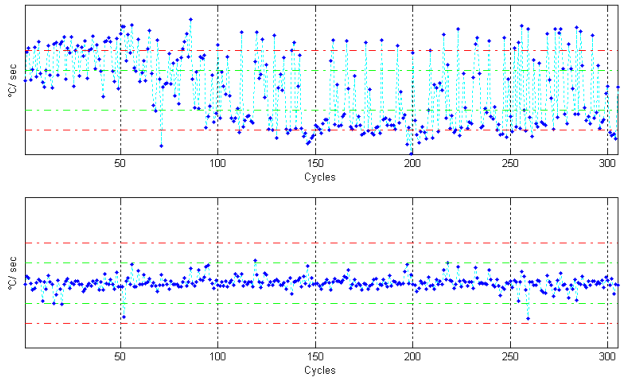

Figure 1: EGT increase at startup, before (top graph) and after normalization (bottom graph) using mainly oil temperature. The top and bottom graphs share the same scales.

The normalization in our application is realized with a general linear model (GLM) that regress each monitored measurement on a space spanned by products and other analytic functions of the context data (inv, sqrt, log, exp…). As the number of such combinations of variables may be really huge, the smallest and best set of variables that will improve a generalized mean square error (MSE) is selected.

For each output variable *r*, each engine *i*, and each snapshot *j*, one models the output value Y_{ij}^r by the linear expression

$$Y_{ij}^r = \mu^r + \alpha_i^r + \lambda_1^r X_{ij}^1 + \cdots + \lambda_q^r X_{ij}^q + \varepsilon_{ij}^r \qquad (1)$$

where each X is one analytic combination of context data, μ is the intercept for output variable r, α is the engine dependency on output variable r (one way to take the engine age into account) and ε is the residual vector. We add also a

constraint $\sum_i n_i \alpha_i^r = 0$ where n_i is the number of flights of engine i to avoid colinearity.

The regression parameters are estimated on a calibration database. But when the number q of input combinations X increases, the quality in term of robustness decreases. The robustness of such algorithm is nothing more than the generalization error. It is computed with a cross-validation method.

Selection of inputs

The lasso criterion captures a good set on inputs. It uses an energy constraint:

$$\arg\min_{\lambda \in R^q} \sum_{i,j} \left(Y_{i,j}^r - \sum_{l=1}^{q} \lambda_l^r X_{i,j}^l \right)^2 \quad \text{where} \quad \sum_{l=1}^{q} \left| \lambda_l^r \right| < C^r \qquad (2)$$

According to equation (2), the coefficients are optimized to minimize the quadratic reconstruction error. The constraint C defined for each output variable r increases progressively, for each value of C a cross validation test is computed to check the mean square error on observations that were not used for model calibration. At first the MSE decreases, but it finally increases when the system begin to over-parameterize. The parameter vector λ that corresponds to this minimal value is chosen. Figure 2 is a representation of the algorithm results for fuel flow normalization. The biggest absolutes values of coefficients corresponds to combinations that where selected.

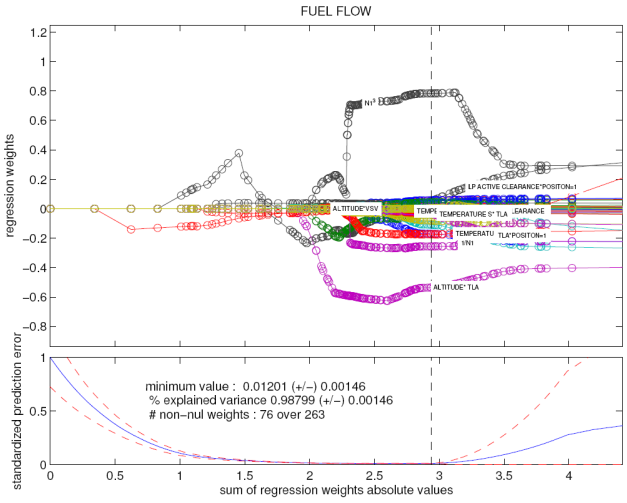

Figure 2: Selection of input combinations X of context measurements that impacts the evolution of fuel flow. The abscisse is the value of the constraint *C*. The top graph gives the coefficients λ. The bottom graph shows the evolution of the MSE. The vertical dashed line points the energie that was selected.

Table 2 below gives some clues about selected variables. In fact up to hundred combinations may be selected for a variable normalization, but most of the coefficients are really small.

Table 2: List of inputs selected for each engine variables. Up to 100 combinations may be selected; this table gives just some elements.

N2	TAT, 1/N1, VSV*XM
FF	$N1^3$, ALT*TLA, 1/N1
T3	TAT, N1, HPTACC, BLD
PS3	TAT*N1, HPTACC, TLA, 1/sqrt(N1)
EGT	TAT, $N1^3$, 1/VBV, XM, TLA^3, 1/TLA

Changes detection and smoothing

Figure 3 is a plot of the initial core speed value and the corresponding computed residual. The jump on the right was completely hidden in the noise; it is discovered only after normalization. Our goal is to classify the input data and a good thing is to get rid of any residual noise (the small residual noise around the jump on the right graph). This noise can be removed by incremental smoothing procedure but some abrupt changes that correspond to maintenance operations may perturb this smoothing. The curve of successive observations is modeled by linear trends. Then at each time a change test detects a jump and if the test is positive the smoothing process is reinitialized.

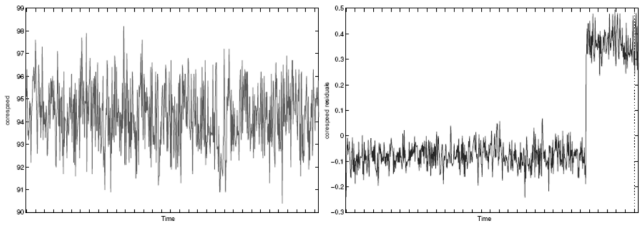

Figure 3: Result of normalization. On the left is the initial core speed (N2) measurement and on the right the residuals of the regression.

The trend estimation use a recursive least squares algorithm (RLS). After initialization flight l_m the trend is estimated until current flight l (see equation 3). The model error is computed and tested according to chosen parameters.

$$\arg\min_{\alpha,\beta} \sum_{j=l_m+1}^{l} \theta^{(l-j)} \left(Y_{i,j}^r - (\beta j + \alpha) \right)^2 \qquad (3)$$

If the test detects a change at flight l_{m+1} the computation is reinitialized. This test can be implemented as a multivariate computation, thus when a change is detected all computation, on each variable, are reinitialized simultaneously. Figure 4 presents a sudden change well observed in temperatures and fuel flow. It is important that this whole process remains iterative, so the new smoothed observations are automatically computed and may be immediately treated by the classification algorithm.

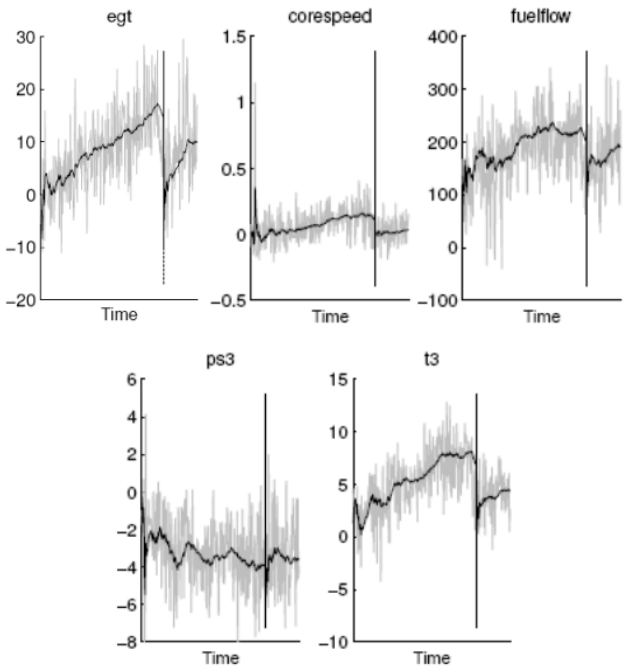

Figure 4: Change detection.

Self-organizing map

SOM (also called Kohonen maps) is a tool often used in data-mining applications to represent high dimensional observations in a lower dimensional space. In our case, the 10=5*2 dimensional input space (5 variables and two snapshots per flight) is projected on a flat 2D space, hence allowing the possibility to plot each engine state on a map. As a dimension reduction solution, SOM can be seen as a kind of non linear principal component analysis (PCA). In the 2D space of the map, an hexagonal grid of K nodes $i = 1, \dots K$ defines a lattice. A set of prototype vectors $\{m_1, \dots, m_K\}$ is associated to each node in the initial space. The autoadaptive process updates the prototypes in the initial space to better represent the data in the 2D space. This learning operation is implemented trought an iterative algorithm and has the interesting property that it "preserves" distances (i.e. observations that are close in the initial space remain close on the 2D map). The algorithm "moves" the prototypes (the lattice nodes) in the initial space. It behaves as if the lattice is deformed in the original domain to cover most of the observations.

Let $\{x_1, \dots, x_N\}$ be the set of observations used to calibrate the map. The algorithm proceeds in two phases and repeat until some stability criterion is reached:

For each step, an observation x_j is selected.

- The competitive phase searches the best matching unit c for observation j: $c = \arg\min_i \left\| x_j - m_i \right\|$.
- Then the cooperative phase moves the BMU but also other close prototypes on the map to the observation x_j according to a neighborhood function h_{ci} defined on the grid (eq. 4 and 5).

Equation (4) details the iterative distortion of the grid over time t. The coefficient α is a step factor that decreases with time to slowly stabilize the process and ensure convergence.

$$m_i(t+1) = m_i(t) + \alpha(t) h_{ci}(t) \left(x_j - m_i(t) \right) \qquad (4)$$

Our selection of distance h affects only close units on the grid:

$$h_{ci}(t) = \exp\left(- d_{ci} / 2\sigma_t \right) \qquad (5)$$

where σ is a radius that defines the neighborhood on the map and decrease with time and d is a chosen distance between nodes on the map.

Figure 5 present the backgrounds (coordinate values of each prototype) for all input variables. This representation gives an understandable interpretation of each map node. For example the top-left corner concentrates the states with high EGT and high FF, the left of bottom-right corner is high N2 but low FF and medium EGT.

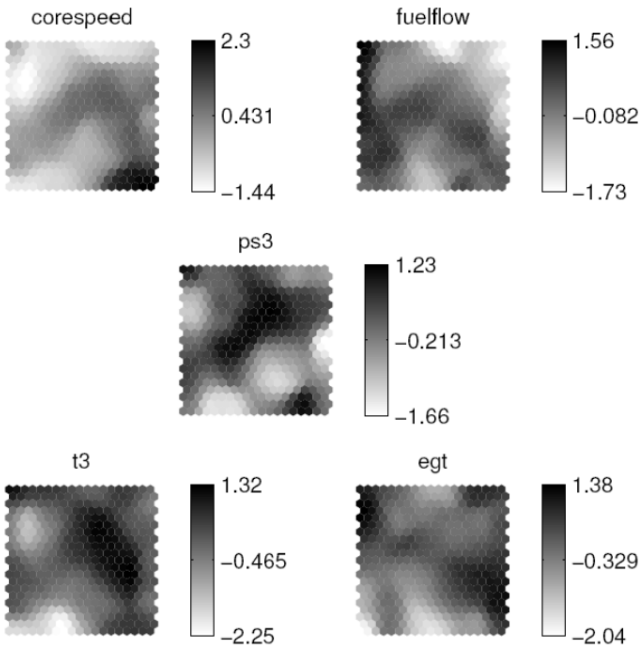

Figure 5: Background representation of a map example.

The map evolves when new data appears so the background representation changes also. The next two figures present a little smoother EGT background over witch the evolution of two engines is superposed.

On each figure the trajectories of an engine is represented by successive dots which size decrease and color goes from red to orange, green, yellow and blue.

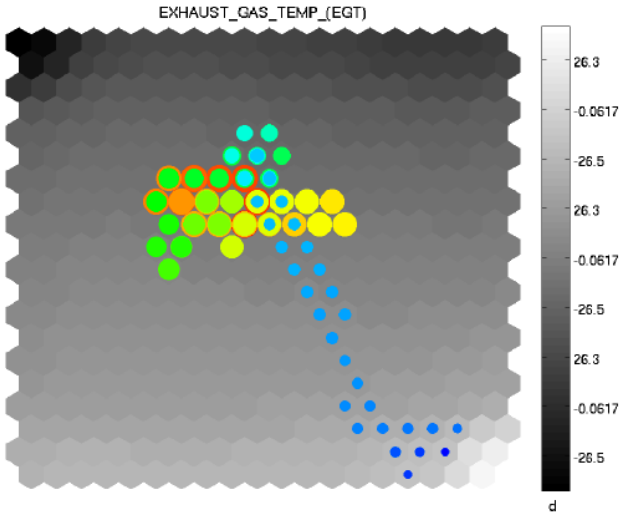

Figure 6: This trajectory (big-red to small blue dots) presents a classical increase of EGT under normal operating conditions.

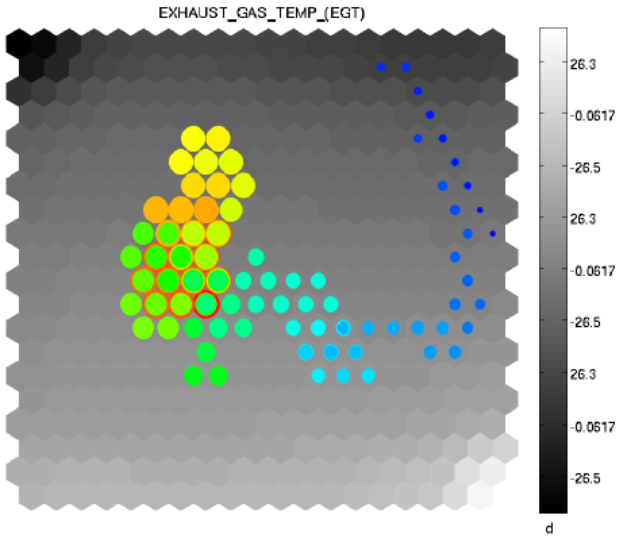

Figure 7: This trajectory is unusual, the EGT begins to increase (which is normal) but then in stabilize and even decrease a little without evident maintenance operation (sudden jump in the trajectory).

Trajectories may have jumps. Figure 8 corresponds to the evolution of an engine state. The numbers 1 to 607 are successive flights. One can observe different parts on this trajectory. Each jump may be a maintenance operation or an engine event. There are different kinds of events: scheduled maintenance or unexpected damages. Most time when such event appears, the configuration of the engine changes and the effect is generally a discontinuity on the trajectory. The jumps on the map correspond to the instants found by the previous change detection algorithm.

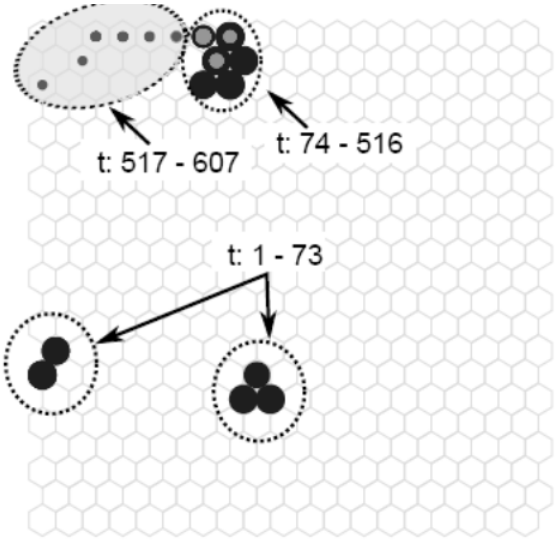

Figure 8: General trajectories, like this one, present jumps that correspond to maintenance operations.

4. APPLICATION

The preceding section describes the kind of mathematical algorithms we develop to analyze the state evolution of an engine in our fleet. Lets now sketch the application we propose to analyze the engine fleet.

Fleet cartography

Select a set of engines, say the ones that belong to some similar planes of a given airline company and realize equivalent missions. Then map the current engine state on Figure 9 map with a different number (say SN) per engine.

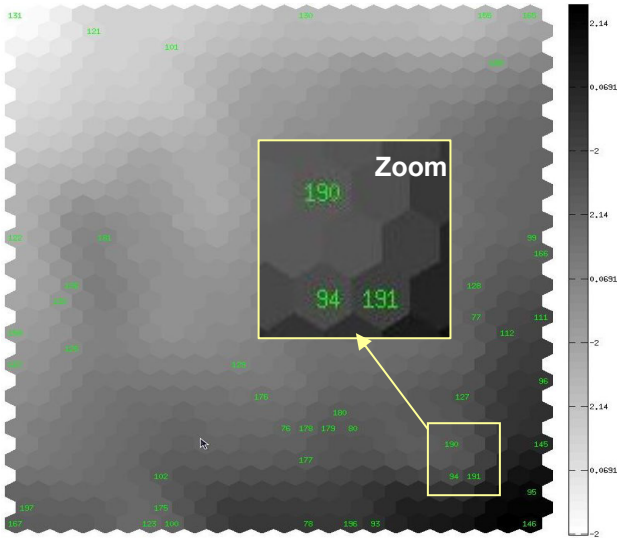

Figure 9: Snapshot of a part of an engine fleet on one map. Each green number is the SN of the engine.

This representation helps to immediately identify a set of engines with high EGT, for example.

Another representation uses a classification of the cells. This is commonly used by SOM applications and generally done by hierarchical clustering (Figure 10).

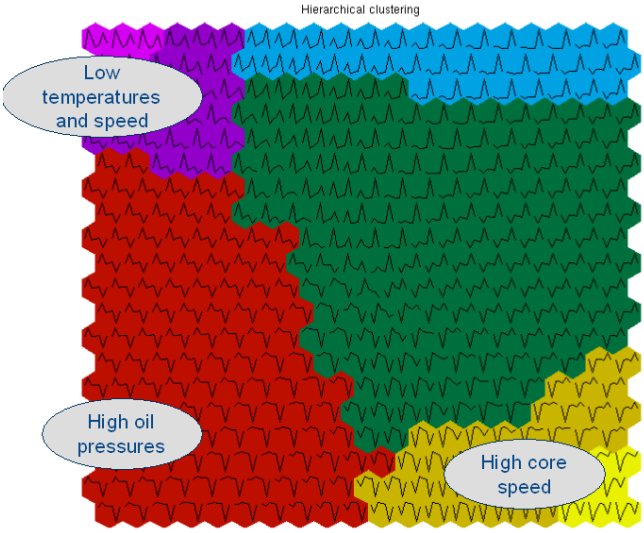

Figure 10: Classification of cells, then definition of labels to identify each part of the map.

This classification may replace the classical background in a representation like the one on Figure 6 and Figure 7.

Statistic queries

As soon as the engine states may be plotted and classified on a 2D lattice, it is possible to compare trajectory parts. For this purpose we use an editing distance (string distance) on the map. Each cell on the map corresponds to a label and the distance between two labels depends on their position on the grid. To compare two strings of labels (two trajectories) we use a cost function. This function gives individual cost for suppression or addition of a label and a replacement cost which depends on the distance on the map between the two labels. This type of distance allows comparing two strings of different sizes.

Suppose that we need to compare two trajectories described by strings s^1 and s^2. Each string is a succession of labels $s^j = [s_1^j, s_2^j, \ldots, s_{n_j}^j]$ where each s_i^j is a position on the map. Suppose now that one can find p operations $o_1, \ldots, o_p$ among suppression, insertion and replacement that transform the chain s^1 into s^2. Then the cost of this global transformation is equal to the sum of all individual costs cost(o_i), and the distance from s^1 to s^2 is defined by the minimum value of such operation (equation 6).

$$d(s^1, s^2) = \min_{o:o(s^1)=s^2} \sum_{i=1}^{\text{length}(o)} \text{cost}(o_i) \qquad (6)$$

Then, using the last flights of an engine (the end of the engine trajectory), seek in the fleet database for similar paths on the map. The nearest paths are part of past trajectories of other engines. An analysis of the future (just after matching with the initial path) leads to clues for the evolution of the current engine. It is possible to count each event that appears during the nearest future of the other engines to build a prognostic. It is even possible to use delays between the current equivalent position and the first event to anticipate a delay of availability before failure. Moreover with the existence of a big fleet database it is possible to compute precision or confidence intervals for our statistics.

Use cases

This methodology was tested in Snecma on a small fleet of 140 engines during one year. It is also used for bearing analysis where the inputs are replaced by multi-scale indicators extracted form order-spectrograms. In fact this same methodology may be adapted for a lot of different purposes and is currently evaluated on civil and military programs to help maintenance organization.

5. CONCLUSIONS

This last application is a serious evolution of datamining tools dedicated to fleet maintenance. The next step is an online prototype implementation that will include a way to correlate the unsupervised classification with a flow of maintenance events. Our goal is to label the hierarchical clustering and add this information to the statistic queries.

With such labels connected to parts of the map, the future statistics will be detailed as probability to encounter a known event.

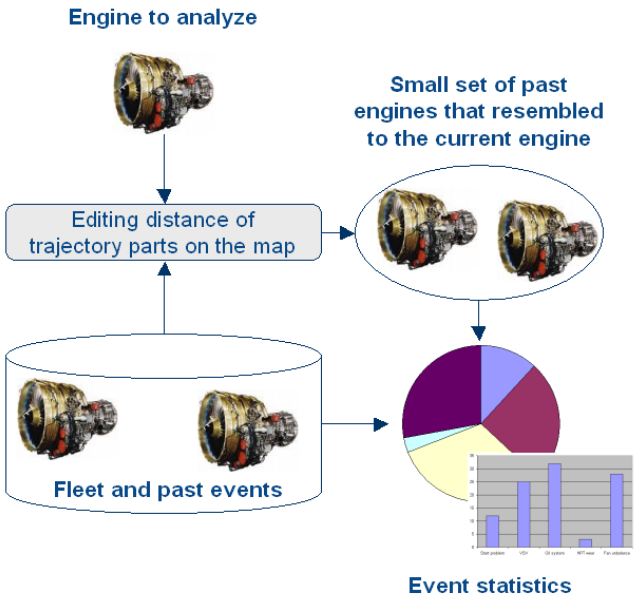

Figure 11: Statistic application of the visual mining methodology.

REFERENCES

[1] M. Basseville, I. Nikiforov, "Detection of Abrupt Changes: Theory and Application". Prentice-Hall (1993).

[2] S. Haykin, "Neural Networks, a Comprehensive Fondation", MacMillan, IEEE Press, 1994.

[3] V. N. Vapnik, "The Nature of Statistical Learning", Springer Verlag, NY, 1995.

[4] T. Kohonen, "Self-Organizing Maps", Springer Series in Information Sciences, Vol 30, Springer, 1995.

[5] F. Gustafsson, "Adaptative filtering and change detetction". John Wiley & Sons (2000).

[6] J. Vesanto, J. Himberg, E. Alhoniemi, J. Parhankangas, "Som toolbox for matlab 5". Technical Report A57, Helsinki University of Technology (2000).

[7] J. Lacaille, « Industrialisation d'algorithmes mathématique », habilitation à diriger des recherches, Université Paris 1, Sorbonne, France, 2004.

[8] B. Efron, T. Hastie, I. Johnstone, R.J. Tibshirani, "Least angle regression". Annals of Statistics 32(2) (2004) 407–499.

[9] J. Lacaille, M. Zagrebnov, "A statistical approach of abnormality detection and its applications", AEC/APC 2006, Denver, CO.

[10] J. Lacaille, "How to automatically build meaningful indicators from raw data", AEC/APC 2007, Palm Spring, CA.

[11] J. Lacaille, M. Zagrebnov, "An Unsupervised Diagnosis for Process Tool Fault Detection: the Flexible Golden Pattern", IEEE Transactions on Semiconductor Manufacturing, Volume 20, Issue 4, Page(s): 355 – 363, 2007.

[12] M. Svensson, S. Byttner, T. Rgnvaldsson, "Self-organizing maps for automatic fault detection in a vehicle cooling system". In: 4th International IEEE Conference on Intelligent Systems. Vol. 3. (2008) 8–12.

[13] M. Cottrell, G. Gaubert, C. Eloy, D. Franois, G. Hallaux, J. Lacaille, M. Verleysen: "Fault prediction in aircraft engines using self-organizing maps". In: Advances in Self-Organizing Maps. Volume 5629, Springer 37–44 (2009).

[14] J. Lacaille, "Standardized Failure Signature for a Turbofan Engine", in Proceedings of IEEE Aerospace Conference 2009, Big Sky, MO.

[15] X. Flandrois, J. Lacaille, et all. "Expertise Transfer and Automatic Failure Classification for the Engine Start Capability System", in Proceedings of AIAA Infotech 2009, Seattle, WA.

[16] J. Lacaille, R. N. Djiki, "Model Based Actuator Control Loop Fault Detection", in Proceedings of Euroturbo Conference 2009, Graz, Austria.

[17] S. Blanchard et all. « Health Monitoring des moteurs d'avions », les entretiens de Toulouse 2009, France.

[18] M. Cottrell et all. "Fault prediction in aircraft engines using Self-Organizing Maps", in Proceedings of WSOM 2009, Miami, FL.

[19] A. Ausloos et all. "Estimation of monitoring indicators using regression methods; Application to turbofan start sequence", ESREL 2009, Prague, Poland.

[20] J. Lacaille, "An Automatic Sensor Fault Detection and Correction Algorithm", AIAA ATIO 2009, Hilton Beach, SC.

[21] J. Lacaille, "A Maturation Environment to Develop and Manage Health Monitoring Algorithms", PHM 2009, San Diego, CA.

[22] G. Ross, D. Tasoulis, N. Adams, "Online annotation and prediction for regime switching data streams". In: Proceedings of ACM Symposium on Applied Computing (2009) 1501–1505.

[23] J. Lacaille. "Validation of Health Monitoring Algorithms for Civil Aircraft Engines". In IEEE Aerospace Conference, Big Sky, MT, 2010.

[24] E. Côme, M. Cottrell, M. Verleysen, and J. Lacaille. "Self Organizing Star (SOS) for Health Monitoring". In ESANN, Bruges, 2010.

[25] E. Côme, M. Cottrell, M. Verleysen, and J. Lacaille. "Aircraft Engine Health Monitoring using Self-Organizing Maps". In ICDM, Berlin, Germany, 2010.

[26] A. Hazan, M. Verleysen, M. Cottrell, and J. Lacaille. "Trajectory Clustering for Vibration Detection in Aircraft Engines". In ICDM, Berlin, Germany, 2010.

[27] H. Hazan, M. Verleysen, M. Cottrell, J. Lacaille, "Linear smoothing of FRF for aircraft engine vibration monitoring", ISMA 2010, Louvain.

[28] J. Lacaille, V. Gerez, R. Zouari, "An Adaptive Anomaly Detector used in Turbofan Test Cells", PHM 2010, Portland, OR.

BIOGRAPHY

Jérôme Lacaille is senior expert in algorithms for Snecma. He joined the company in 2007 with responsibility for developing a health monitoring solution for jet engines. Jérôme has a PhD from the Ecole Normale Supérieure, France in Mathematics. Jérôme has held several positions including scientific consultant and professor. He has also co-founded the Miriad Technologies Company, entered the semiconductor business taking in charge the direction of the Innovation Department for Si Automation (Montpellier - France) and PDF Solutions (San Jose - CA). He developed specific mathematic algorithms that where integrated in industrial process. Over the course of his work, Jérôme has published several papers on integrating data analysis into industry infrastructure, including neural methodologies and stochastic modeling.

Etienne Côme is researcher in the French National Institute for Transport and safety Research. He worked in collaboration with Snecma as a post-doc in the SAMM laboratory of Sorbone, Paris 1.